

In the Claims:

1. (Currently Amended) In a network data multiple routing system comprised of a plurality of paths of interconnectable router nodes, each router node controlled by software processing and management information for enabling data routing along a predetermined path of each of the plurality of router nodes between common input and output terminal nodes for the paths, a method of revising and upgrading the software information in one of such preselected router nodes along one of said paths for each of the multiple routers, that comprises, continuing the data routing along said one path between said terminal nodes with original software information controlling said one router node; during such continued routing, preparing new software information at said one router node from said original software information and including revisions and upgrades while said one router node continues routing data between said terminal nodes; and, after such preparing of the new software information within said one router node, swapping the ~~same~~ proposed new software information for the original software information in said one router node during ~~the~~ its continuing of the data routing along said one path without any internal or external interruption and even if no alternative router paths are available in the network, and with the swapping effected imperceptibly to all the other router nodes in the multiple path router system.
2. (Original) The method of claim 1, wherein the original software package contains a border gateway protocol (BGP) for controlling data packet routing from said one node along the network, and said preparing of the new software includes registering and binding the original software BGP information in a new software package during continued data packet routing through said one node under the original software BGP.
3. (Original) The method of claim 2 wherein said revisions and upgrades are also bound into said new software package.
4. (Currently Amended) The method of claim 3 wherein the original software package is signaled by the new software package when it is prepared to take over the software control of said one node, and thereupon ~~pre-emptively~~ pre-emptively swaps with the original software package in said node without interruption.
5. (Original) The method of claim 4 wherein prior to the swap, the original software package checks and prepares for its disconnection at the time the new software package activates its swap connection.

6. (Currently Amended) A network data multiple routing system having, in combination, a plurality of paths of interconnected router nodes each router node controlled by software processing and managing information for enabling data routing along a predetermined path of each of the plurality of router nodes between common input and output terminal nodes for the paths; apparatus for revising and upgrading the software information in one of the router nodes along one of said paths for each of the multiple routings comprising means operable during the continued data routing along said one path; between said terminal nodes, for preparing new software information at said one router node from said original software information within said one router node and including revisions and upgrades while said one router node continues routing data between said terminal nodes; and, means operable after such preparing of the new software information within said one router node, for swapping the ~~same~~ prepared new software information for the original software information in said one router node during the continuing of the data routing along said one path without any internal or external interruption, and even if no alternative router paths are available in the network, the swapping means enabling said swapping imperceptibly to all the other router nodes in the multiple path routing system.

7. (Original) The system of claim 6 wherein the original software package contains a border gateway protocol (BGP) for controlling data packet routing from said one node along the network, and said means for preparing said new software includes means for registering and binding original software BGP information in a new software package during continued data packet routing through said one node under the original software BGP.

8. (Original) The system of claim 7 wherein said binding means also binds the revisions and upgrades into said new software package.

9. (Original) The system of claim 8 wherein, means is provided for enabling the new software package to signal when it is prepared to take over the software control of said one node, and means for thereupon pre-emptively swapping with the original software package in said node without interruption.

10. (Original) The system of claim 9 wherein, prior to the swap, means is provided for enabling the original software package to check and prepare for its disconnection at the time the new software package activates its swap connection.

11. (Original) The system of claim 10 wherein an IP handler for all IP is provided, interconnected with the original software package and the revised software package BGPs for controlling each of the registering, binding and pre-emptive swapping means.

12. (Original) To the system of claim 11 wherein a software backplane architecture is provided for enabling the operation of the swapping means comprising three stacks: a control and data methods interface, active router module interfaces, and backplane services with which the modules interact and reply upon.

C/ 13. (Original) The system of claim 12 wherein the backplane services stack contains interacting command line interpreter (CLI), simple network management protocol (SNMP), and hypertext transfer protocol (HTTP) units.

14. (Original) The system of claim 13 wherein the active router module interface stack comprises, in addition to the module, a persistent and shared data interface receiving data from the module and from the data methods interface.

15. (Currently Amended) The system of claim 14 wherein the module is further connected to a management information base (~~MBI~~) (MIB) serving as the control interface to the module, with the module informing the management information base of the ways in which it can be configured; and said backplane services stack further containing a configuration manager for such configuring.

16. (Original) The system of claim 15 wherein said management information base is connected to receive requests from said simple network management protocol unit for the module.

17. (Original) The system of claim 16 wherein the backplane services stack is also provided with a dynamic linker that is connected with the module to permit linking and then replacement with a link to a new version.

18. (Original) The system of claim 16 wherein the backplane services stack is further provided with a task manager for controlling the starting, stopping and handling of information from the old original software and the new upgrade software, and to allow a new task to take control of said task manager to pick up processing where the previous task left off.

19. (Original) The system of claim 18 wherein the backplane services stack is also provided with a dynamic binder that also allows a task to request other tasks and receives a pointer to an interface structure to capture information.

c/ 20. (Original) The system claimed in claim 19 wherein said persistent interface allows information on the system state to be passed from one task to the next, coordinating with IP and messaging interface so that the point where the new task begins processing queued requests, corresponds to the point where the shared persistent data is checkpointed, and with the SNMP/MIB descriptors and data persisting between activation of the service.

21. (Original) The system claimed in claim 19 wherein said task manager controls activation of each required module, monitors their availability, and stops a task or restarts the task if it has crashed.

22. (Original) The system claimed in claim 19 wherein a task activation table is provided that indicates what is running in the system, with said binder controlling what versions are loaded into the system and runnable, and with said task manager controlling which particular software version is running at a given time while monitoring the original and new software swapping.

---